DESARROLLO DE APLICACIONES JAVA

INTRODUCCIÓN

- Java es un lenguaje de programación orientado a objetos. Gracias a la máquina virtual hace que nuestro sistema sea portable y así poder desarrollarlo independiente del sistema operativo donde se ejecuta.
- Consta de:
  - JVM: Máquina virtual, actúa como interprete de los códigos de bytes generados por el compilador sean transformadas a instrucciones comprensibles para la cpu.
  - JRE: Contiene la máquina virtual, además de librerías y utilidades.
  - JDK: Contiene el JRE, además del compilador y herramientas de desarrollo. Existen 3 tipos:
    - JSE: Destinado a aplicaciones de escritorio.
    - JEE: Destinada a aplicaciones web empresariales.
    - JME: Destinado principalmente para aplicaciones de dispositivos móviles.

SINTAXIS BÁSICA. TIPOS BÁSICOS.

- int : Entero 32 bits
- long : Entero 64 bits
- float : Reales 32 bits
- double : Reales 64 bits
- boolean : true o false
- char : Carácter único Unicode de 16 bits
- byte : 8 bits
- short : 16 bits

SINTAXIS BÁSICA. OPERACIONES BÁSICAS.

- Operaciones matemáticas:
  - +, -, /, *
  - Operador módulo a % b : Devuelve el resto de a/b
  - Comparadores <, >, ==
- Operaciones lógicas:
  - && : Conjunción lógica
  - || : Disyunción lógica
  - ! : Negación

LISTAS ESTÁTICAS. ARRAY

- Son listas de elementos de un mismo tipo, con un tamaño fijo.
- Sintaxis : Tipo[] variable;
- Inicialización :
  - Tipo[] variable = {elemento1,elemento2…elementoN};
  - Tipo[] variable = new Tipo[MAX];variable[0]=elemento1;… variable[n]=elementoN;
- El atribute length nos da el tamaño del Array.
- Mátriz : Es un Array de Array. Tipo[][] matriz;

COMPARADORES

- if(condicion){…} else{…}:
  - Si se cumple la condición se ejecuta lo que hay en el primer bloque sino ejecuta el segundo bloque de instrucciones.
  - Se puede ir anidando varios ELSE IF.
  - Tiene una forma abreviada: condicion?... : ... ;
- switch(value):
  - Comprueba el valor y en función de su valor ejecuta el bloque cuyo CASE tiene el valor, por ejemplo case 5: ... break; dentro del case no se abren llaves y para indicar que se termina hay que poner el break sino ejecuta el siguiente CASE hasta encontrar un break o no tener más casos.

BUCLES

- for : Tiene la siguiente sintaxis for(inicialización;condición;operación de incremento o decremento)
  - Se suele usar para recorrer listas, se repite hasta cumplir la condición, en la operación o step se define cuál es la operación sobre el contador a realizar en cada iteración.
- while : Tiene la siguiente sintaxis while(concición)
  - Se ejecuta mientras cumple la condición.
- do{ … }while(condición);
  - Se ejecuta siempre una vez al menos y se itera tantas veces hasta que deje de cumplir la condición.

TIPOS DINÁMICOS. STRINGS

- Son objetos que guardan cadenas de caracteres, tienen métodos y utilidades para trabajar con ellas. Se puede ver una documentación completa en https://docs.oracle.com/javase/7/docs/api/java/lang/String.html
- Métodos más comunes:
  - + , concat : Concatena Strings.
  - compareTo : Lo compara con otro String.
  - equals : Devuelve true si es igual al String comparado.
  - indexOf : Devuelve la posición de un substring siendo -1 sino existe.
  - contains : Devuelve true si el String a comparar es parte del mismo.
  - toLowerCase : Transforma a minúsculas.
  - toUpperCase : Transforma a mayúsculas.
  - substring : Devuelve un substring entre las posiciones indicadas.
  - replace , replaceAll : Remplazan un carácter por otro, en caso de ser All reemplaza todas las ocurrencias.

TIPOS DINÁMICOS. ARRAYLIST.

- Es una Clase muy utilizada cuando no sabes cuantos elementos va a contener tu lista, ya que se pueden ir añadiendo elementos sin necesidad de definir un tamaño.
- Sintáxis : ArrayList<T> lista = new ArrayList<T>();
- Métodos más utilizados:
  - add : Añade un elemento a la lista, se puede especificar la posición donde añadir el elemento.
  - size : Devuelve el tamaño de la lista;
  - remove : Elimina un elemento de una posición dada.
  - contains : Indica si la lista contiene un elemento.
  - indexOf : Devuelve la posición de un elemento, en caso de no existir devuelve -1;
- Se puede consultar todos los métodos en https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

TIPOS DINÁMICOS. OTROS

- Existen muchas Clases que nos pueden servir de utilidad como:
  - HashMap<K,V> https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html
  - LinkedList<T> https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
  - Stack<T> https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html
  - Vector<T> https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html

CLASES Y OBJETOS

- Las clases son fragmentos de código, donde se definen atributos, métodos y funciones.
- Dichos atributos pueden ser public, protected, private. Donde public pueden ser llamados desde cualquier parte que tenga acceso a una instancia de la clase, protected solo tiene acceso aquellas clases que extiende de la clase que lo define y private solo se puede acceder en la clase.
- Un objeto es una instancia o llamada de una Clase no abstracta. Una clase abstracta es aquella que implementa una interfaz o extiende de otra clase abstracta y no implementa todos los métodos definido por la interfaz.
- Si un objeto es asignado a otro objeto, ambos pasan a apuntar la misma información y si modificas uno u otro el otro se ve afectado por los mismos cambios. Si no se desea este comportamiento hay que clonarlo. Al igual si pasas un objeto por referencia en algún método, toda modificación del mismo en le método será efectiva también en el objeto introducido, es decir los cambios no se quedan en ámbito local.

HERENCIA Y POLIMORFISMO

- La herencia es utilizada cuando queremos que una clase extienda de otra y así heredar todos los atributos y métodos que sean public y protected.
- Una vez una clase extiende de otra, se puede redifinir método existentes, pudiendo llamar al funcionamiento del método de la clase heredada con el uso de super.{nombre método}()
- Si se quiere que una clase tenga unos métodos a definir de manera obligatoria se hace uso de interface para que la clase lo implemente, es usado comúnmente en Oyentes, aunque se puede usar más casos.
- Si una clase queremos que implemente algunos métodos pero no todos y ya otras clases se encarguen de implementarlos para su caso, se tienen que definir como abstractas.

MVC

- Patrón de diseño de software, con el que separamos nuestro sistema en:
  - Modelo: Contiene todo lo que tiene que ver con el modelo de datos y utilidades.
  - Vista: Es donde se definen todos los componentes visuales, donde solo se implementa aquello que tiene que ver con lo visual.
  - Controlador: Es el encargado recibir y controlar eventos, casi siempre sobre la vista. Por otro lado también se encarga de llamar al modelo para procesar la información solicitada y de enviar el resultado a la vista, es decir que también hace de intermediario entre el modelo y la vista.